# NETCONF Agent

- About the NETCONF Agent, on page 1
- Guidelines and Limitations for NETCONF, on page 3
- Configuring the NETCONF Agent, on page 3
- Using the NETCONF Agent, on page 4
- Troubleshooting the NETCONF Agent, on page 8

## About the NETCONF Agent

The Cisco NX-OS NETCONF Agent is a client-facing interface that provides secure transport for the client requests and server responses in the form of a YANG model, encoded in XML.

The NETCONF Agent supports a candidate configuration feature. The Candidate configuration datastore temporarily holds candidate configuration and any changes you make without changing the running configuration. You can then choose when to update the configuration of the device with the candidate configuration when you commit and confirm the candidate configuration.

If you do not confirm the changes, exit from a nonpersistent NETCONF client session, or choose to cancel the commit after you commit the change, a system timer then times out and rolls back the changes if you do not confirm the changes.

If you initiate a confirmed-commit operation with a persistent token, the NETCONF client session becomes a persistent process. In a persistent process, exiting the NETCONF client session will not call an automatic roll-back and the changes cannot be rolled back without the matching persistent token.

Cisco NX-OS NETCONF supports the following configuration capabilities:

- Writable-Running Capability

  ```
  urn:ietf:params:netconf:capability:writable-running:1.0
  ```

- Rollback-on-error Capability

  ```
  urn:ietf:params:netconf:capability:rollback-on-error:1.0
  ```

- Candidate Configuration Capability

  ```
  urn:ietf:params:netconf:capability:candidate:1.0
  ```

- Validation Capability

  ```
  urn:ietf:params:netconf:capability:validate:1.1
  ```

- Confirmed Commit Capability

```
    urn:ietf:params:netconf:capability:confirmed-commit:1.1
```

When a new session starts, the NETCONF Agent sends out a <hello> message advertising its capabilities. The following example shows a NETCONF agent sending a <hello> message to the client:

```
<?xml version="1.0" encoding="UTF-8"?>
<hello>

 <capabilities>

  <capability>urn:ietf:params:netconf:base:1.0</capability>
  <capability>urn:ietf:params:netconf:base:1.1</capability>
  <capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
  <capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
  <capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
  <capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
  <capability>urn:ietf:params:netconf:capability:confirmed-commit:1.1</capability>

<capability>http://cisco.com/ns/yang/cisco-nx-os-device?revision=2017-04-06&amp;module=cisco-nx-os-device&amp;deviations=cisco-nx-os-device-deviations</capability>

 </capabilities>

 <session-id>1438752697</session-id>
</hello>
```

The Cisco NX-OS NETCONF Agent supports the following NETCONF Protocol operations:

- get
- get-config
- edit-config
- close-session
- kill-session

Candidate configuration supports the following NETCONF Protocol operations.

- Operations for the candidate configuration as <source> or <target>.
    - get-config
    - edit-config
    - copy-config
    - lock
    - unlock
    - validate
- Operations for the candidate configuration that do not require explicitly specifying the candidate configuration as <source> or <target>.
    - commit
    - cancel-commit
    - discard-changes

**Note** The delete-config operation is not allowed.

## Guidelines and Limitations for NETCONF

The NETCONF Agent has the following guideline and limitation:

- The device YANG model defines ephemeral data and they are marked with a comment "// Ephemeral data". These nonpersistent large-volume data is handled differently from the rest of the model. They are returned only when `<get>` query's `<filter>` parameter points specifically to the particular element marked with the comment. Refer to the ephemeral data support documentation for detailed information on the usage.
- In a single Get request, the number of objects that are supported is 250,000. If you see the following error, it means that the data requested is more than 250,000. To avoid this error, send requests with filters querying for a narrower scope of data.

  ```
  too many objects(459134 > 250000) to query the entire device model.
  ```
- NETCONF does not support enhanced Role-Based Access Control (RBAC) as specified in RFC 6536. Only users with a "network-admin" role are granted access to the NETCONF agent.
- The `<edit-config>` "replace" operation sometimes might not work due to run-time default values and behaviors that are implemented by the affected system component. Therefore, it's better to base the configuration to replace on the configuration obtained through the `<get-config>` query instead of the NX-API Developer Sandbox.
- The Cisco NX-OS NETCONF server supports a maximum of five subscriptions, one subscription per client session.
- Per RFC 5277, autonomous notifications support NETCONF, SYSLOG, and SNMP streams for event sources. In this release, Cisco NX-OS supports NETCONF streams only.
- Cisco NX-OS does not support the Replay option for subscriptions. Because Start Time and Stop Time options are part of Replay, they are not supported.
- For a stream subscription and filtering, support is only for subtree filtering. XPath filtering is not supported.
- When the Cisco NX-OS NETCONF Agent is operating under a heavy load, it is possible that some event notifications can get dropped.
-

## Configuring the NETCONF Agent

The NETCONF Agent supports the following optional configuration parameters under the `[netconf]` section in the configuration file (`/etc/mtx.conf`).

| Parameter | Description |
|---|---|
| **idle_timeout** | (Optional) Specifies the timeout in minutes after which idle client sessions are disconnected.<br>The default value is 5 minutes.<br>A value of 0 disables timeout. |
| **limit** | (Optional) Specifies the number of maximum simultaneous client sessions.<br>The default value is 5 sessions.<br>The range is 1 to 10. |

The following is an example of the `[netconf]` section in the configuration file:

```
[netconf]
mtxadapter=/opt/mtx/lib/libmtxadapternetconf.1.0.1.so
idle_timeout=10
limit=1
```

For the modified configuration file to take effect, you must restart the NETCONF Agent using the CLI command [**no**] **feature netconf** to disable and re-enable.

# Using the NETCONF Agent

### General Commands

The NETCONF Agent is enabled or disabled by the command [**no**] **feature netconf**.

### Initializing the Candidate Configuration Datastore

The candidate configuration can only be initialized with the contents of the running configuration. To Initialize the candiate configuring datastore, send a Copy-Config request using SSH, with candidate as the target and running as the source.

### Performing Read and Write on the Candidate Configuration

To read from the candidate configuration, send a Get-Config request with SSH, using candidate as the source.

To write to the contents of the candidate configuration, send an Edit-Config request with SSH, using candidate as the target.

### NETCONF Candidate Configuration Workflow

The candidate configuration workflow is as follows:

- Edit the candidate configuration file.
- Validate the candidate configuration.
- Commit the changes to the running configuration.

### Example: An SSH Session

This example shows initiating a session using the SSH client and sending an Edit-Config and Get request using the SSH Client.

```
client-host % ssh -s admin@172.19.193.152 -p 830 netconf
client-host % ssh -s admin@172.19.193.152 -p 830 netconf
User Access Verification
Password:
<?xml version="1.0" encoding="UTF-8"?>
<hello>
    <capabilities>
        <capability>urn:ietf:params:netconf:base:1.0</capability>
        <capability>urn:ietf:params:netconf:base:1.1</capability>
        <capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
        <capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
        <capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
        <capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
        <capability>urn:ietf:params:netconf:capability:confirmed-commit:1.1</capability>

<capability>http://cisco.com/ns/yang/cisco-nx-os-device?revision=2017-04-06&amp;module=cisco-nx-os-device&amp;deviations=cisco-nx-os-device-deviations</capability>

    </capabilities>
    <session-id>1912037714</session-id>
</hello>
]]>]]><hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.1</capability>
  </capabilities>
</hello>
]]>]]>
#794
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
     <edit-config>
         <target>
             <running/>
         </target>
         <config>
             <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
                     <bgp-items>
                         <inst-items>
                             <dom-items>
                                 <Dom-list>
                                     <name>default</name>
                                        <rtrId>2.2.2.2</rtrId>
                                 </Dom-list>
                             </dom-items>
                         </inst-items>
                     </bgp-items>
             </System>
         </config>
     </edit-config>
</rpc>
##

#190
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
```

```
</rpc-reply>

##

#511
<rpc message-id="109"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get-config>
    <source>
        <running/>
    </source>
    <filter type="subtree">
        <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
                <bgp-items>
                    <inst-items>
                        <dom-items>
                            <Dom-list/>
                        </dom-items>
                    </inst-items>
                </bgp-items>
        </System>
    </filter>
</get-config>
</rpc>
##

#996
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="109"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <System>
            <bgp-items>
                <inst-items>
                    <dom-items>
                        <Dom-list>
                            <name>default</name>
                            <always>disabled</always>
                            <bestPathIntvl>300</bestPathIntvl>
                            <holdIntvl>180</holdIntvl>
                            <kaIntvl>60</kaIntvl>
                            <maxAsLimit>0</maxAsLimit>
                            <pfxPeerTimeout>30</pfxPeerTimeout>
                            <pfxPeerWaitTime>90</pfxPeerWaitTime>
                            <reConnIntvl>60</reConnIntvl>
                            <rtrId>2.2.2.2</rtrId>
                        </Dom-list>
                    </dom-items>
                </inst-items>
            </bgp-items>
        </System>
    </data>
</rpc-reply>

##
```

Note that the operation attribute in `edit-config` identifies the point in configuration where the specified operation will be performed. If the operation attribute is not specified, the configuration is merged into the existing configuration data store. Operation attribute can have the following values:

- create
- merge

- delete

The following example shows how to delete the configuration of interface Ethernet 0/0 from the running configuration.

```
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
        <target>
            <running/>
        </target>
        <default-operation>none</default-operation>
        <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
            <top xmlns="http://example.com/schema/1.2/config">
                <interface xc:operation="delete">
                    <name>Ethernet0/0</name>
                </interface>
            </top>
        </config>
    </edit-config>
</rpc>]]>]]>
```

## Error Messages

If a request results in an error, the response payload includes the error.

### Errors Defined by Cisco

The following are the errors defined by Cisco.

| Error defined by Cisco | Description |
|---|---|
| unknown-error-cond | Unknown error encountered. |
| n-y-i | The requested operation is not supported. (not-yet-implemented). |
| namespace-not-found | Error in request payload. |
| namespace-already-exists | Error in request payload. |
| object-not-found | Error in request payload. |
| object-not-container | Error in request payload. |
| object-not-property | Error in request payload. |
| no-property-in-object | Error in request payload. |
| invalid-dn | Internal error. |
| invalid-arg | Internal error. |
| already-exists | Error in request payload. |
| container-not-found | Error in request payload |
| container-already-exists | Error in request payload. |
| property-not-found | Error in request payload. |
| property-already-exists | Error in request payload. |

| | |
|---|---|
| malformed | Error in request payload. |
| alloc-failed | Internal error. |
| sigint | Internal error. |
| not-initialized | Internal error. |
| inappropriate | Internal error. |

The following is an example of a NETCONF error response payload that reports an invalid IP address value:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="320" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>Protocol</error-type>
    <error-tag>operation-failed</error-tag>
    <error-severity>Error</error-severity>
    <error-message xml:lang="en">Property Merge (set property) Failed: operation-failed
value=500.500.500.500</error-message>
    <error-path>/config/System/bgp-items/inst-items/dom-items/Dom-list/rtrId</error-path>
  </rpc-error>
</rpc-reply>
```

# Troubleshooting the NETCONF Agent

### Troubleshooting Connectivity

- From a client system, ping the management port of the switch to verify that the switch is reachable.
- In the bash shell of the switch, execute the **service netconf status** command to check the agent status.
- There is the XML Management Interface (also known as xmlagent), which is quite different from and often confused as the NETCONF Agent. Please ensure that you connect to the correct port 830 and receive a correct <hello> message (similar to what is shown in the Establishing a NETCONF Session section) from the server if the server does not respond with the correct NETCONF messages.
- You can view NETCONF agent debugs from the Bash shell by viewing the contents of the /volatile/netconf-internal-log file. You can enable the Bash shell by using the **feature bash** command. After enabling the Bash shell, enter the Bash shell through the **run bash** command. For more information, see the chapter titled *Bash* in this document.

  **Note:** The **debug netconf** commands cannot be used to debug NETCONF Agent operations. These debug commands will not output any NETCONF Agent-related logs.